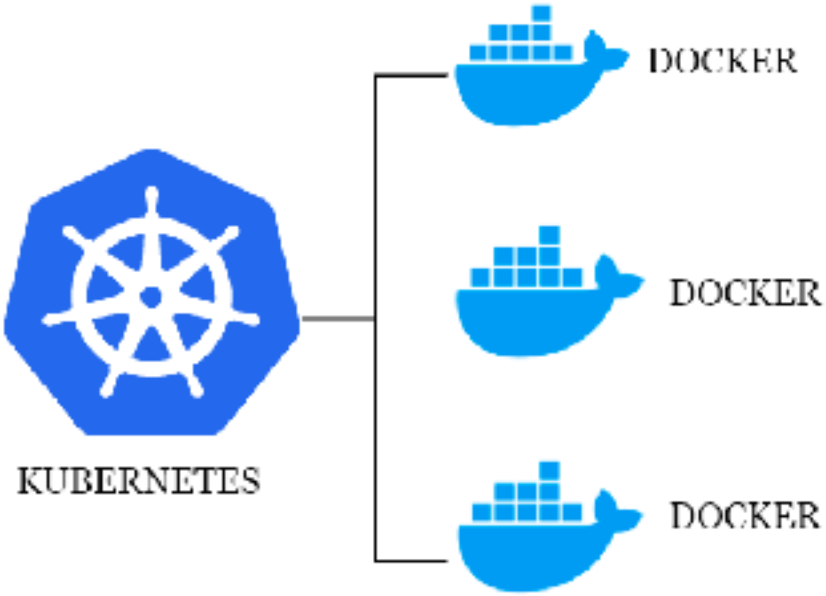
Docker & K8S

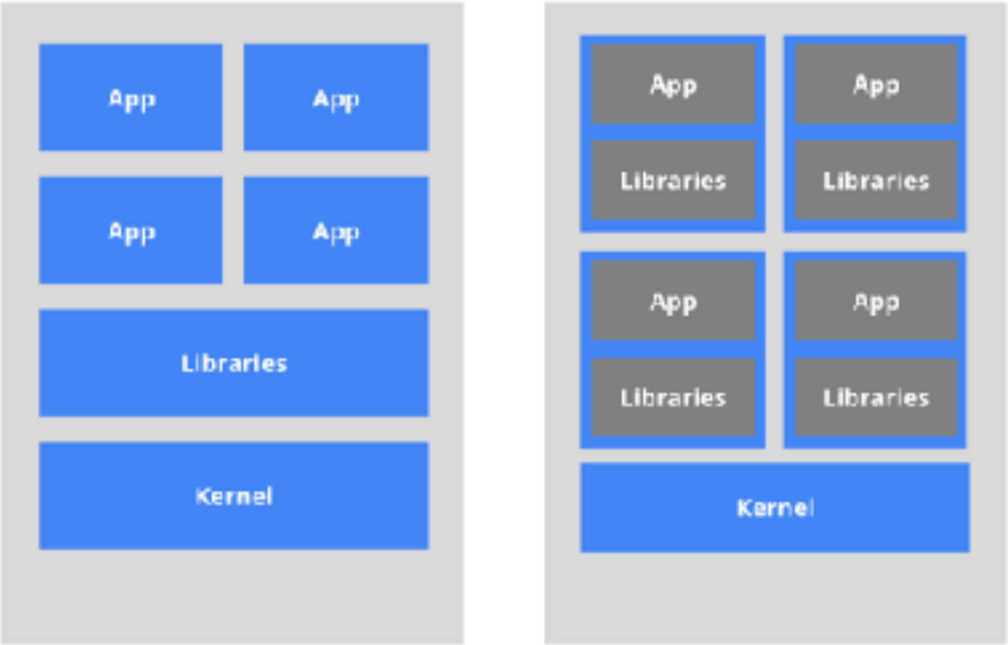
Thibault SAUSSAC



What's containers ?

The old way: Applications on host

The new way: Deploy containers



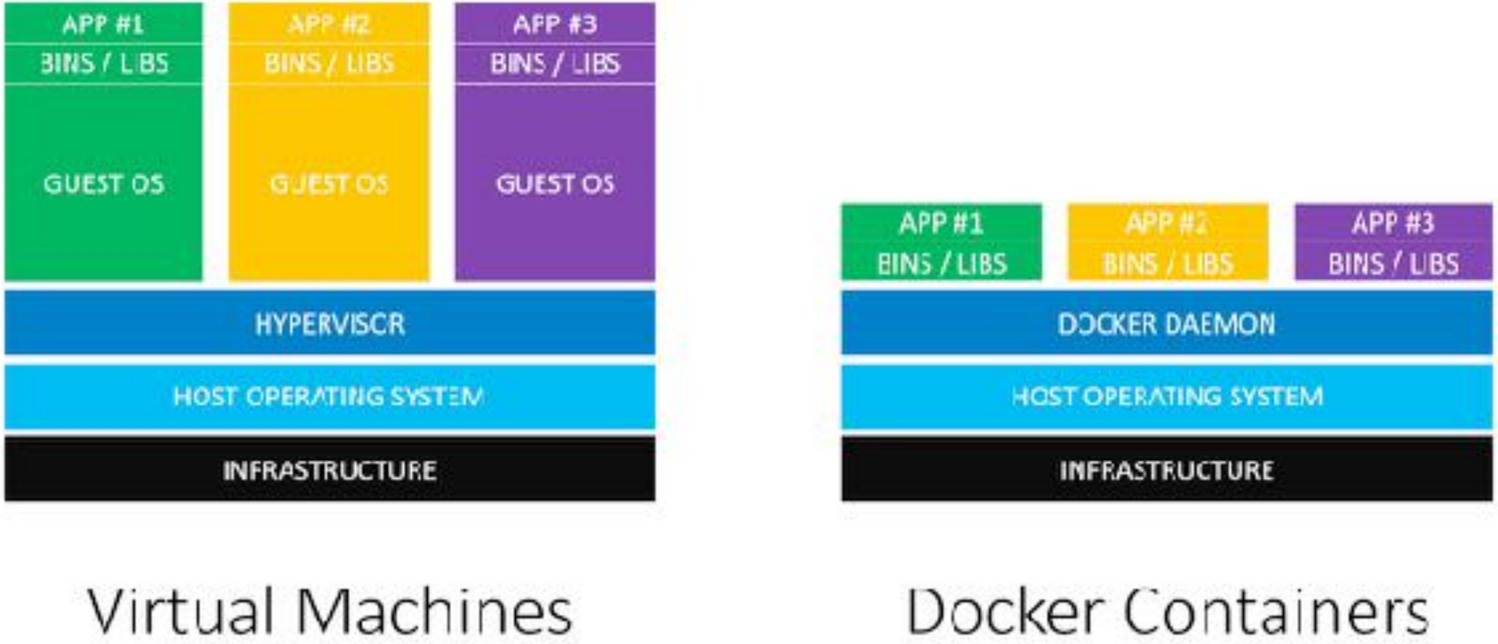
Heavyweight, non-portable
Relies on OS package manager

Small and fast, portable
Uses OS-level virtualization

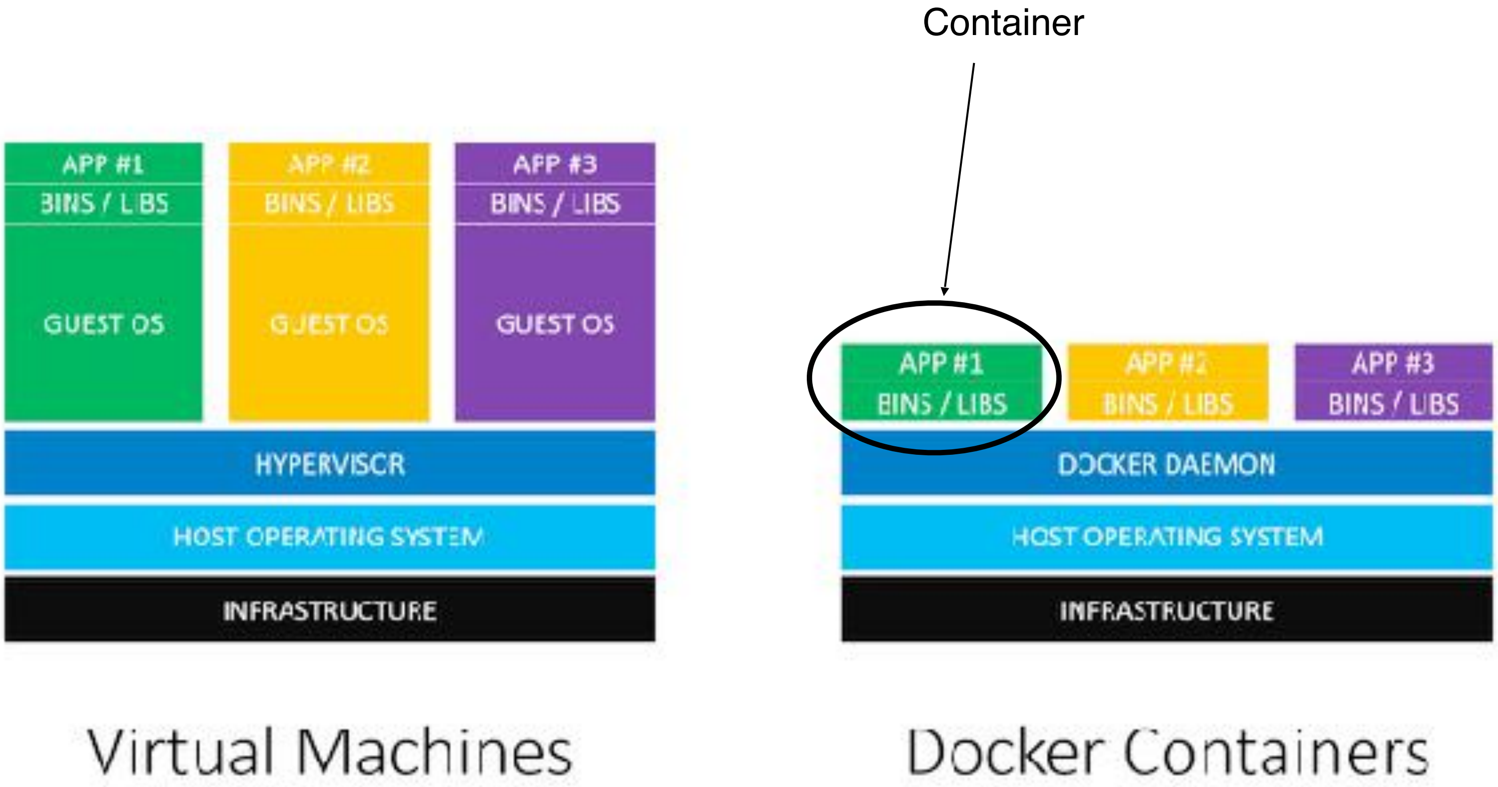
img : kubernetes.io

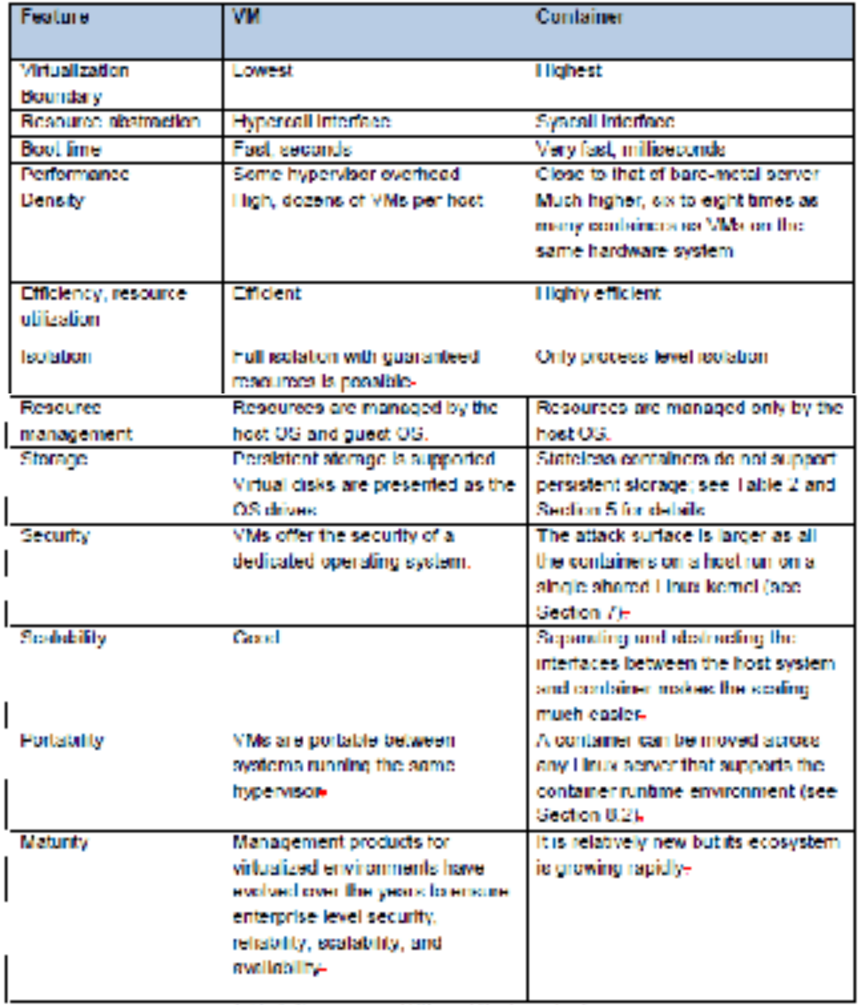
What's Docker ? (vs virtual machines)

Les conteneurs Docker peuvent être considérés comme des VM sans leur propre système d'exploitation
Ils partagent tous le système d'exploitation hôte et sont séparés à l'aide d'outils du kernel tels que cgroups, namespaces, ..
Du point de vue de conteneur, chaque conteneur a son propre système de fichiers.



What’s Docker ? (vs virtual machines)





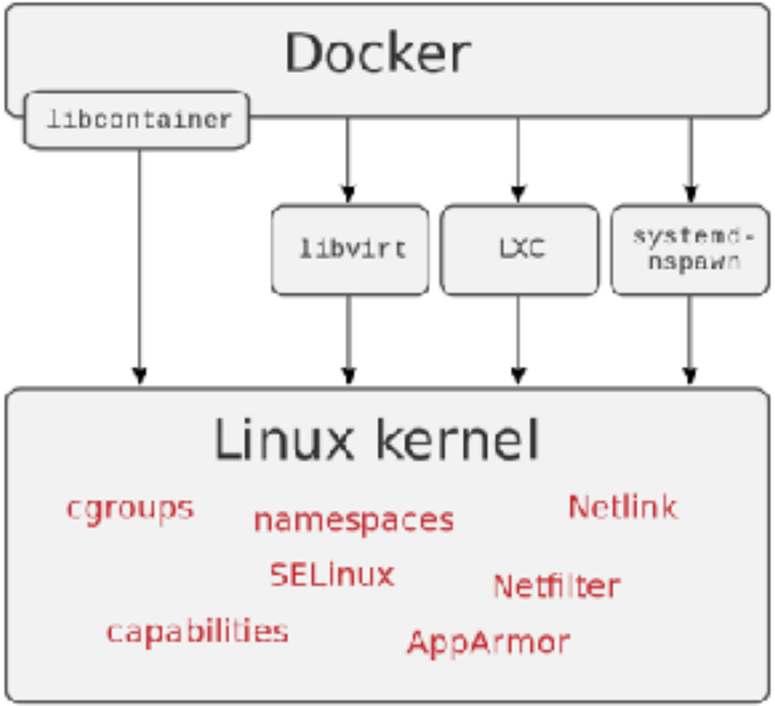
Feature	VM	Container
Virtualization Boundary	Lowest	Highest
Resource abstraction	Hypercall interface	Syscall interface
Boot time	Fast, seconds	Very fast, milliseconds
Performance	Some hypervisor overhead	Close to that of bare-metal server
Density	High, dozens of VMs per host	Much higher, six to eight times as many containers as VMs on the same hardware system
Efficiency, resource utilization	Efficient	Highly efficient
Isolation	Full isolation with guaranteed resources is possible.	Only process level isolation
Resource management	Resources are managed by the host OS and guest OS.	Resources are managed only by the host OS.
Storage	Persistent storage is supported Virtual disks are presented as the OS drives	Stateless containers do not support persistent storage; see Table 2 and Section 5 for details
Security	VMs offer the security of a dedicated operating system.	The attack surface is larger as all the containers on a host run on a single shared Linux kernel (see Section 7).
Scalability	Good	Separating and abstracting the interfaces between the host system and container makes the scaling much easier.
Portability	VMs are portable between systems running the same hypervisor.	A container can be moved across any Linux server that supports the container runtime environment (see Section 8.2).
Maturity	Management products for virtualized environments have evolved over the years to ensure enterprise level security, reliability, scalability, and availability.	It is relatively new but its ecosystem is growing rapidly.

Table 1: Comparison of VMs and Container Features

Docker vs. Virtual machines

What's Docker ?

Les interfaces docker avec le kernel Linux

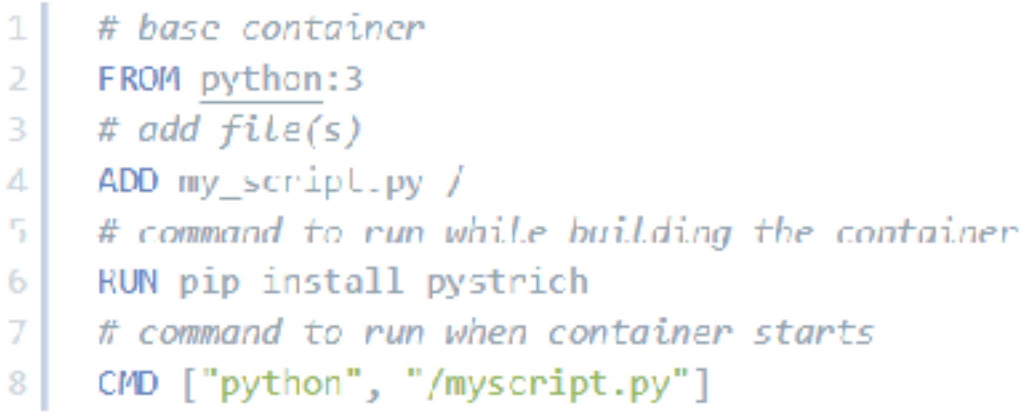


img : https://fr.wikipedia.org/wiki/Docker_(logiciel)

How it works ?

Dockerfile : La recette de cuisine

Un fichier qui décrit comment on build le conteneur

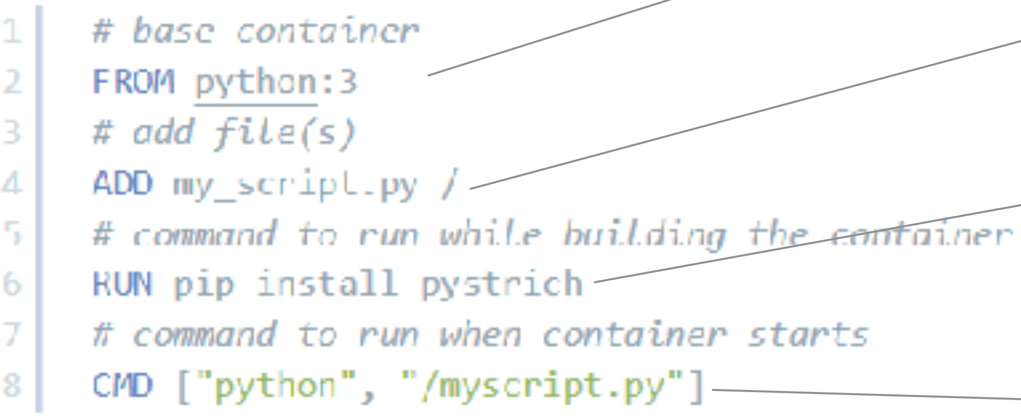


```
# base container
FROM python:3
# add file(s)
ADD my_script.py /
# command to run while building the container
RUN pip install pystrich
# command to run when container starts
CMD ["python", "/myscript.py"]
```

Et la commande, « `docker build` » pour build l'image

How it works ?

Dockerfile : La recette de cuisine



```
# base container
FROM python:3
# add file(s)
ADD my_script.py /
# command to run while building the container
RUN pip install pystrich
# command to run when container starts
CMD ["python", "/myscript.py"]
```

Start from an existing image

Ajouter un ficher dans à la racine

RUN une commande
(Installation des dep python)

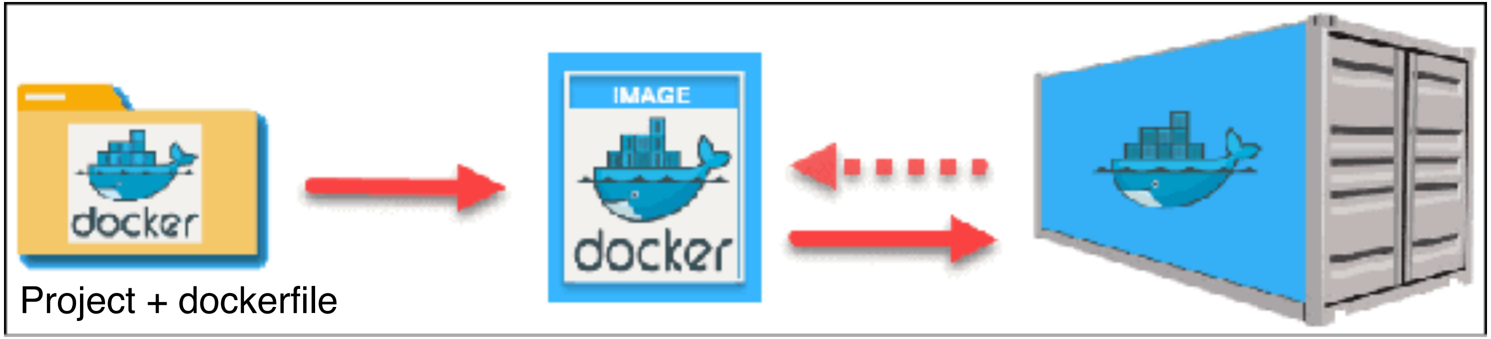
La commande qu’on lance lorsque le Conteneur démarre

How it works ?

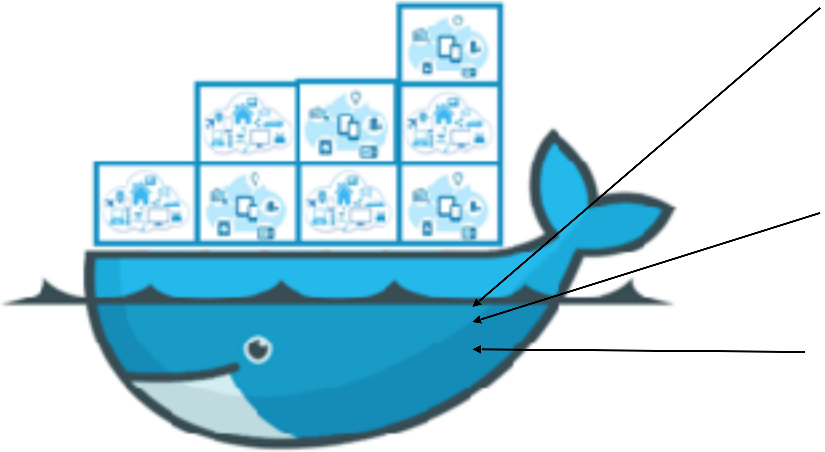
Images vs Containers :

Image: fichier immuable contenant le code source, les bibliothèques, les dépendances, les outils et autres fichiers nécessaires à l'exécution d'une application.

Conteneur: environnement d'exécution virtualisé (en cours d'exécution)



How it works ?



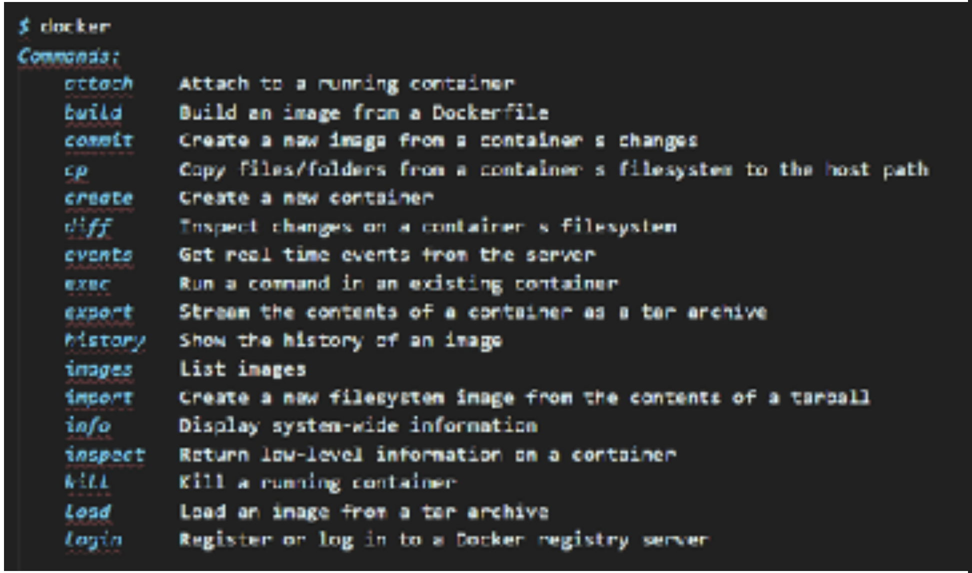
Containers Mgt tools

Images Mgt tools

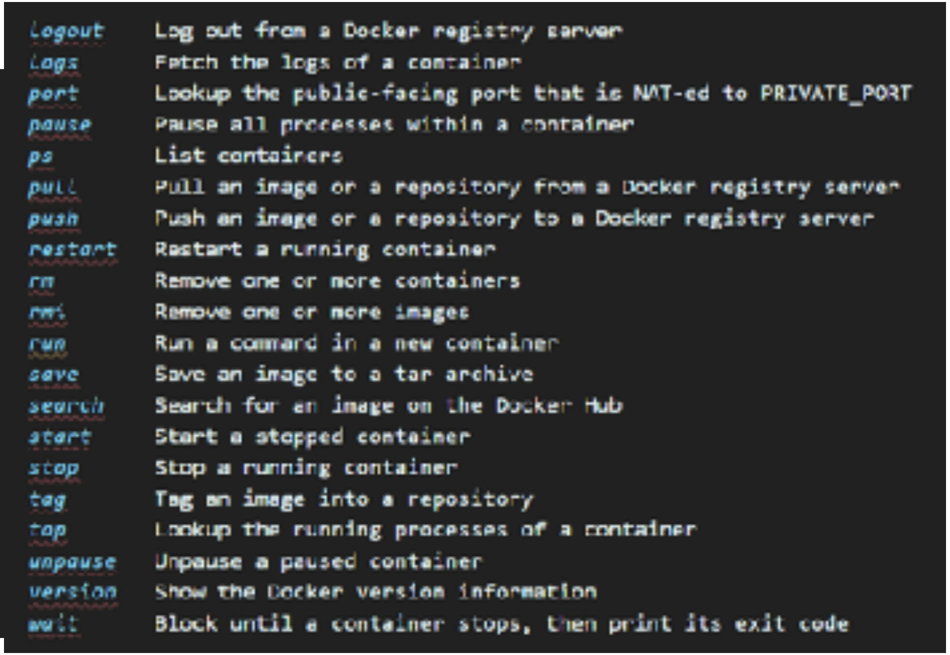
Container Networking tools

img : https://www.lemondeinformatique.fr/

How it works ?



```
$ docker
Commands:
    attach     Attach to a running container
    build      Build an image from a Dockerfile
    commit     Create a new image from a container s changes
    cp         Copy files/folders from a container s filesystem to the host path
    create     Create a new container
    diff       Inspect changes on a container s filesystem
    events     Get real time events from the server
    exec       Run a command in an existing container
    export     Stream the contents of a container as a tar archive
    history    Show the history of an image
    images     List images
    import     Create a new filesystem image from the contents of a tarball
    info       Display system-wide information
    inspect    Return low-level information on a container
    kill       Kill a running container
    load       Load an image from a tar archive
    login      Register or log in to a Docker registry server
```



```
logout     Log out from a Docker registry server
logs       Fetch the logs of a container
port       Lookup the public-facing port that is NAT-ed to PRIVATE_PORT
pause      Pause all processes within a container
ps         List containers
pull       Pull an image or a repository from a Docker registry server
push       Push an image or a repository to a Docker registry server
restart    Restart a running container
rm         Remove one or more containers
rmi        Remove one or more images
run        Run a command in a new container
save       Save an image to a tar archive
search     Search for an image on the Docker Hub
start      Start a stopped container
stop       Stop a running container
tag        Tag an image into a repository
top        Lookup the running processes of a container
unpause    Unpause a paused container
version    Show the Docker version information
wait       Block until a container stops, then print its exit code
```

How it works ?

Nb:

Le conteneur s’arrête si le processus s’arrête.
Si tu veux lancer une image Debian et rien d’autre il est necessaries d’explicité run bash :

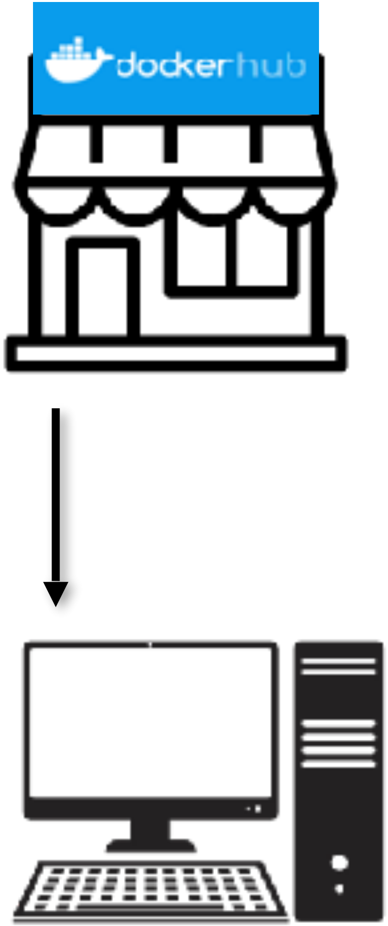
```
docker run -it debian /bin/bash
```

(-i « run interactively » , -t « allocate a pseudo-tty »)

Une fois que le conteneur est arrêté, vous perdez les données à l’interieur.
En cas de besoin de persistance, 2 options : mount a host’ folder dans le conteneur, ou ajouter un « Docker volume ».

The registry

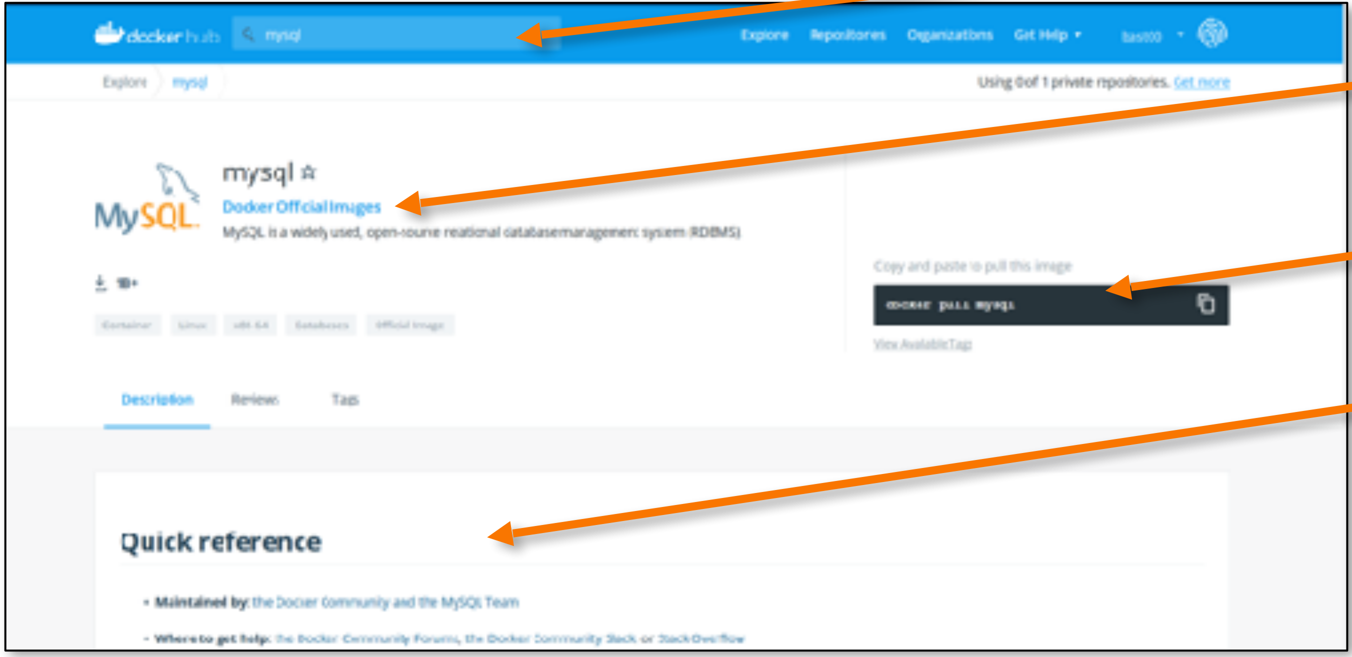
Retrieve images



docker pull [imagename]

imgs : Flaticon / Docker HUB

The registry *hub.docker.com*



Using 0 of 1 private repositories. Get more

mysql

Docker Official Images

MySQL is a widely used, open-source relational database management system (RDBMS).

10M+

Container Linux x86-64 Databases Official Image

Copy and paste to pull this image

```
docker pull mysql
```

View Available Tags

Quick reference

- **Maintained by:** the Docker Community and the MySQL Team
- **Where to get help:** the Docker Community Forums, the Docker Community Slack, or Stack Overflow

Search

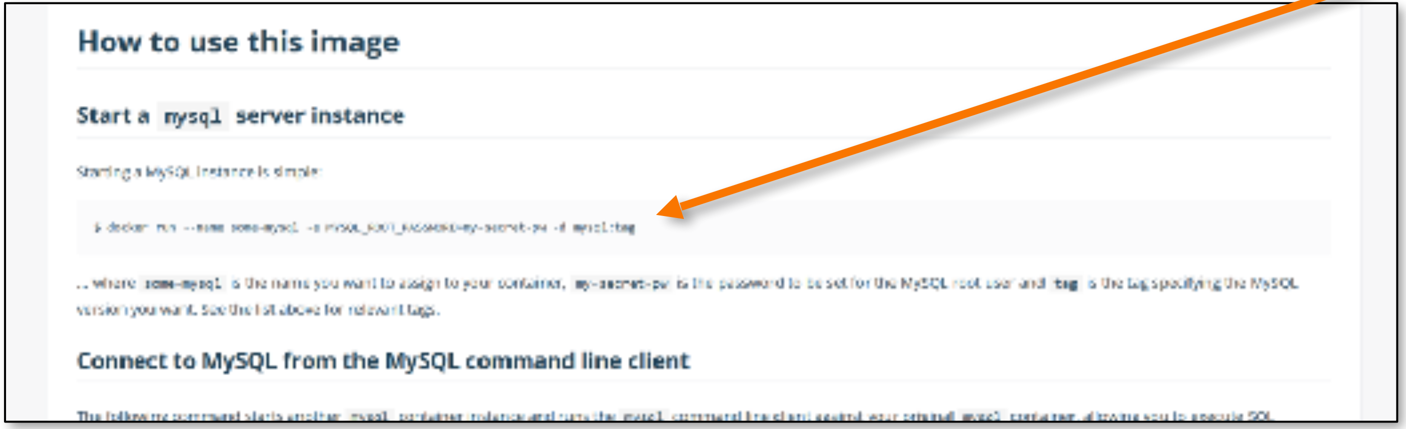
Official image

Download the image

Documentation

img : Docker HUB

The registry *hub.docker.com*



How to use this image

Start a `mysql` server instance

Starting a MySQL instance is simple:

```
$ docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag
```

... where `some-mysql` is the name you want to assign to your container, `my-secret-pw` is the password to be set for the MySQL root user and `tag` is the tag specifying the MySQL version you want. See the list above for relevant tags.

Connect to MySQL from the MySQL command line client

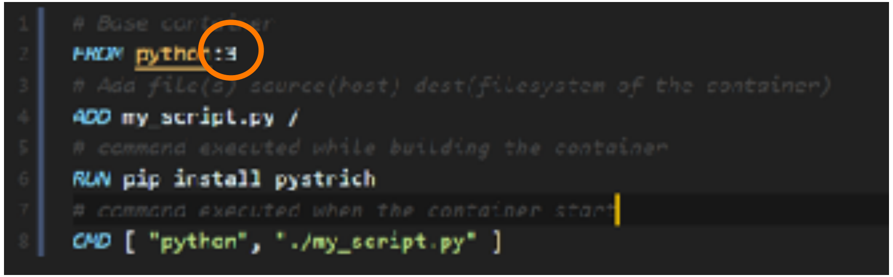
The following command starts another `mysql` container instance and runs the `mysql` command line client against your original `mysql` container, allowing you to execute SQL

Quick setup

Parametre parameters :

- name: nom du conteneur
- d: en background (detach)
- e: variable d'env

The registry *hub.docker.com*



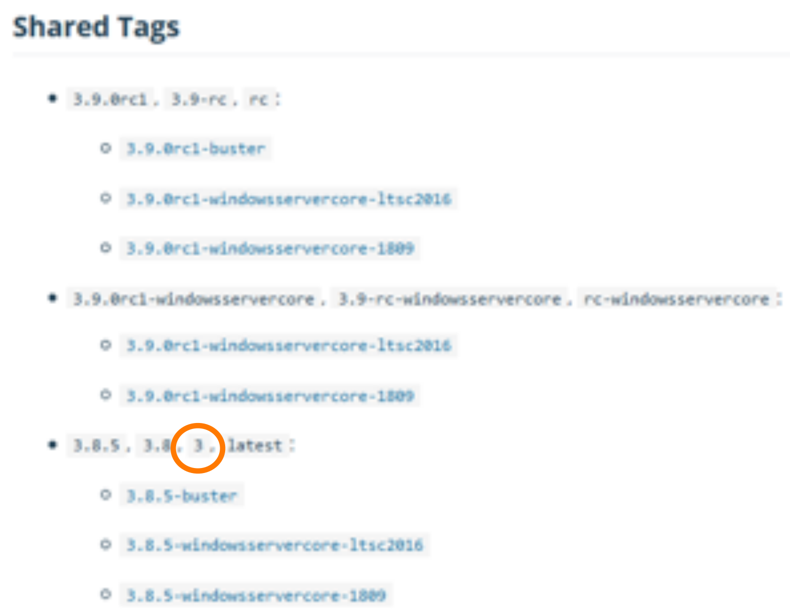
```
# Base container
FROM python:3
# Add file(s) source(host) dest(filesystem of the container)
ADD my_script.py /
# command executed while building the container
RUN pip install pystrich
# command executed when the container start
CMD [ "python", "./my_script.py" ]
```

Précédemment nous utilisions une image de base python:3.

Cela signifie qu'on utilise l'image docker avec le tag 3 ce qui correspond à « 3.8.5-buster »

L'image correspondante est automatiquement téléchargé depuis docker hub quand tu build ton image.

https://hub.docker.com/_/python



Shared Tags

- 3.9.0rc1 , 3.9-rc , rc :
    - 3.9.0rc1-buster
    - 3.9.0rc1-windowsservercore-ltsc2016
    - 3.9.0rc1-windowsservercore-1809
- 3.9.0rc1-windowsservercore , 3.9-rc-windowsservercore , rc-windowsservercore :
    - 3.9.0rc1-windowsservercore-ltsc2016
    - 3.9.0rc1-windowsservercore-1809
- 3.8.5 , 3.8 , 3 , latest :
    - 3.8.5-buster
    - 3.8.5-windowsservercore-ltsc2016
    - 3.8.5-windowsservercore-1809

img : Docker HUB

The registry *hub.docker.com*

Push your own images :

FREE:

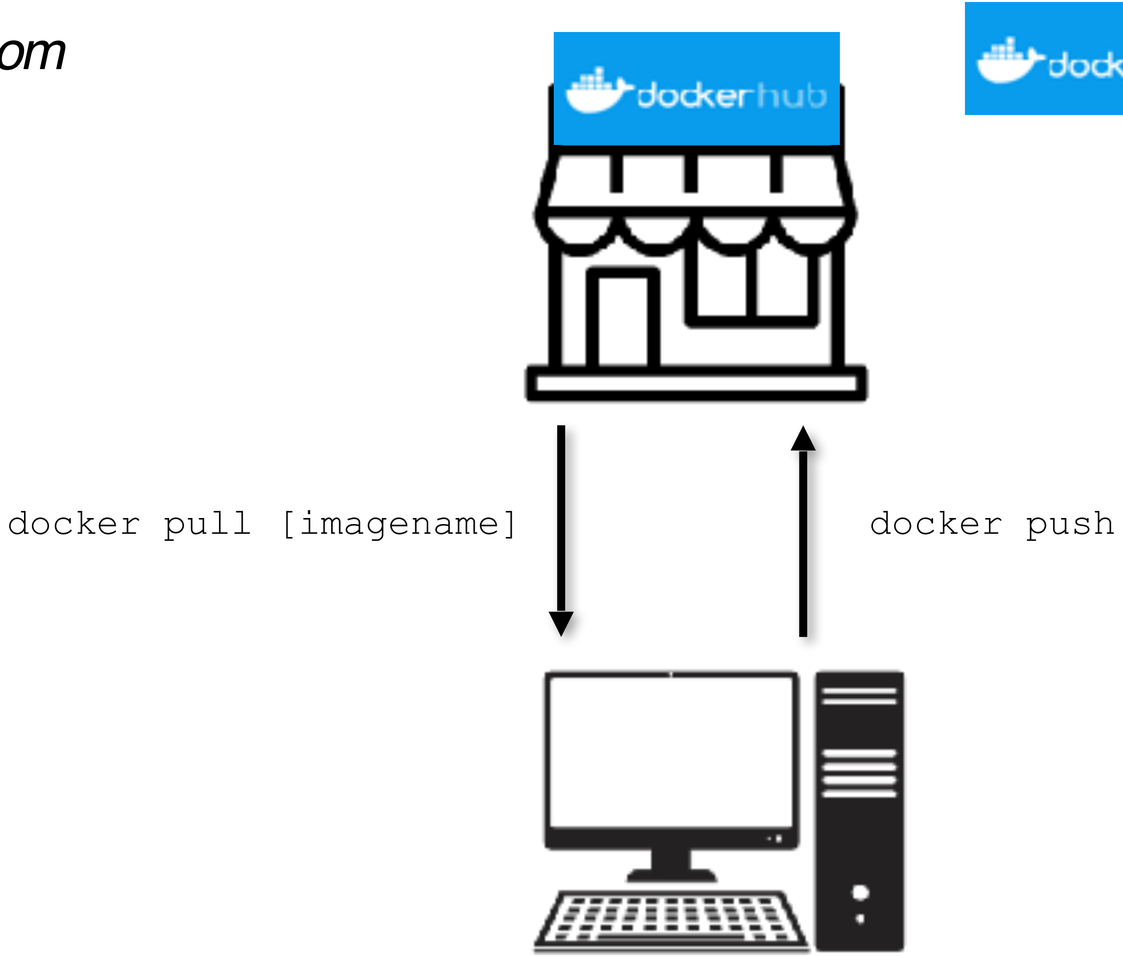
- *Unlimited public repos*
- *One private repo*

PRO (individuals): 5$/months

- *Unlimited private repo*
- *Pro tools for developers*

TEAM: 7$/user/month

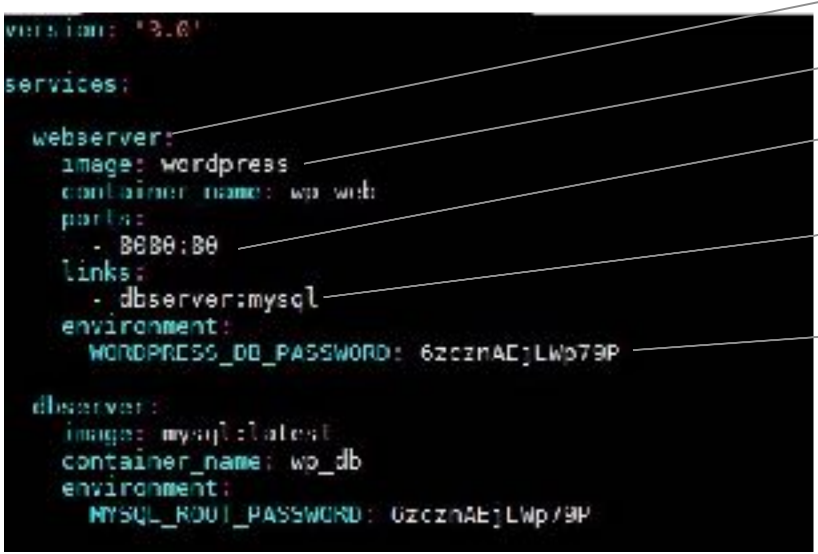
- *Collaboration & Mgt tools*



```
docker pull [imagename]
```

```
docker push
```

Prod : how to setup the prod application : ★★



```
version: '3.0'

services:

  webserver:
    image: wordpress
    container_name: wp_web
    ports:
      - 8080:80
    links:
      - dbserver:mysql
    environment:
      WORDPRESS_DB_PASSWORD: 6zcznAEjLWp79P

  dbserver:
    image: mysql:latest
    container_name: wp_db
    environment:
      MYSQL_ROOT_PASSWORD: 6zcznAEjLWp79P
```

Le service s’appel « webserver »

L’image s’appel « wordpress »

Map le port de l’host 8080 vers le port du conteneur 80

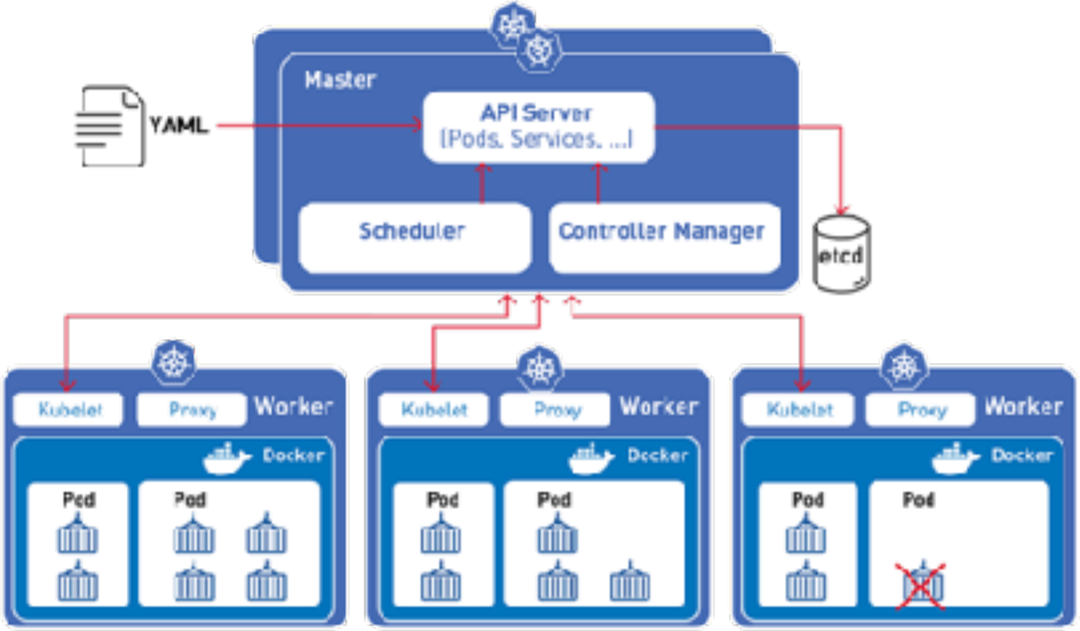
service dbserver ira chercher mysql

Environment variable

Pour utiliser ce fichier:
« docker-compose up »

Il ira automatiquement télécharger WP et MySQL

Kubernetes



- Les hosts sont des « *nodes* »

- Groupe de conteneurs dans les « *pods* »

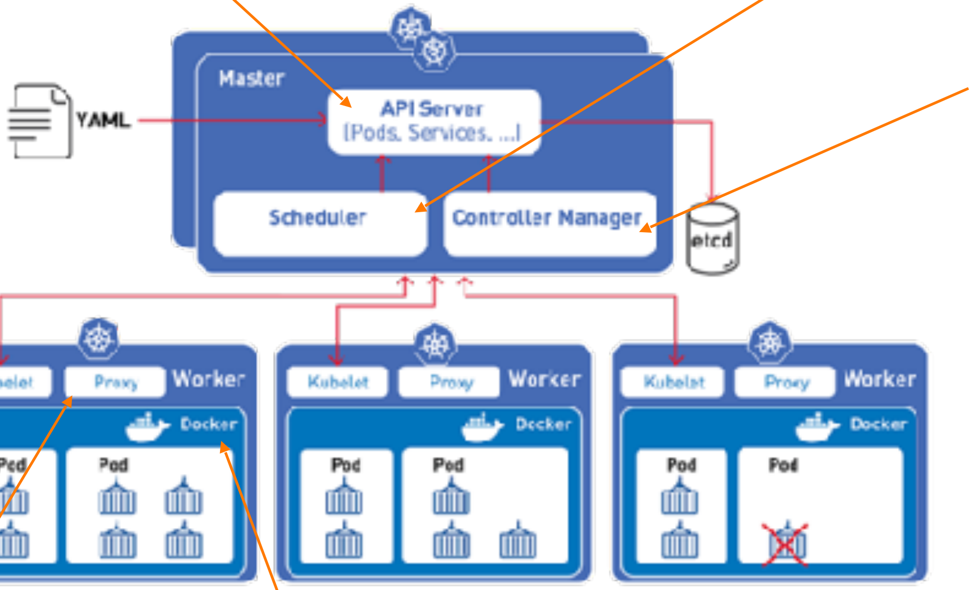
- Plusieurs pods par nodes

- Containers in pods :
  - Shared storage
  - Shared unique cluster IP address

img : https://www.gekko.fr/kubernetes-son-architecture/

Kubernetes

Master node, 3 processes:
- kube-apiserver
- kube-controller-manager
- kube-scheduler



Entrypoint / API

Assign pods to nodes

Control the system

Connect to master

Network services

Container runtime (e.g. Docker)

Nodes, 2 processes:
- kubelet
- kube-proxy

& container runtime

Kubernetes : How to use

CLI : kubectl

Dashboard

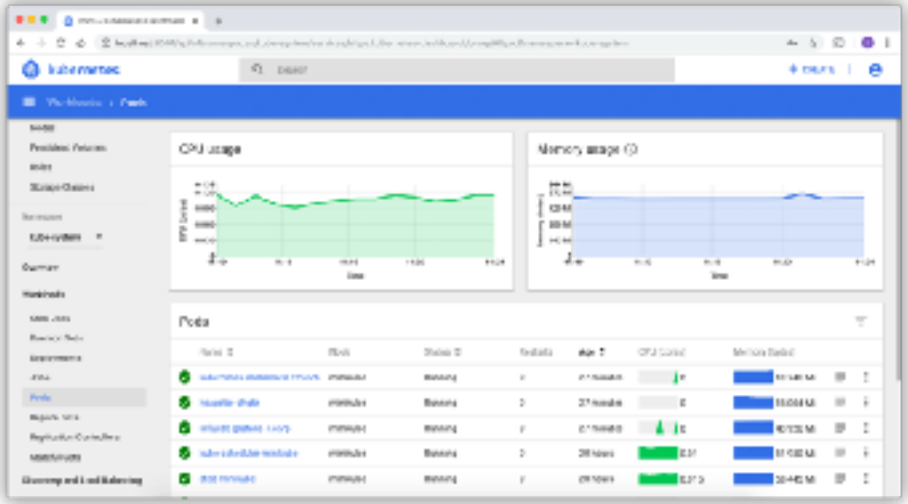
```
kubectl [action] [resource]
```

e.g.

`kubectl get pods` – list the pods

`kubectl get nodes` – list the nodes

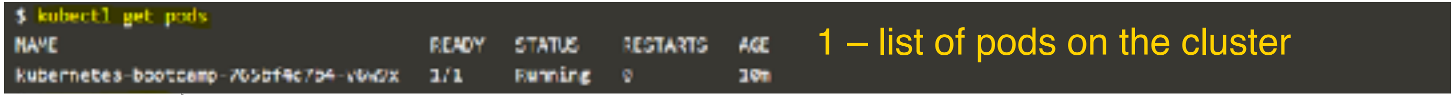
```
kubectl create deployment mydeploy \
--image=rabbitmq:latest
```



Créera un « *deployment* » qui appellera un conteneur basé sur RabbitMQ dispo sur Docker Hub.
Par défaut il crée 1 container dans 1 pod.

img : kubernetes.io

Kubernetes : How to use - Example



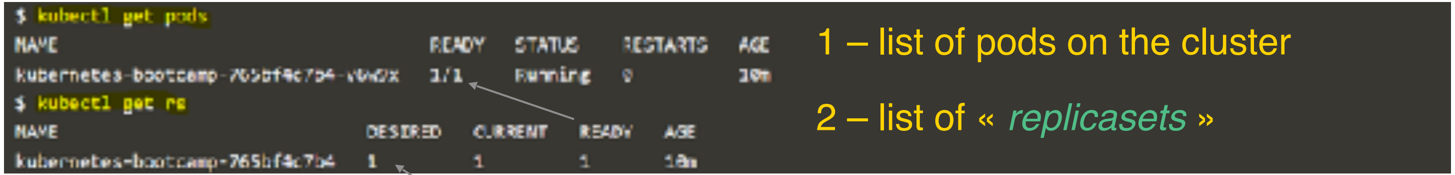
```
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-v6w2x  1/1     Running   0          10m
```

1 – list of pods on the cluster

Info : This container is the result of a deployment (here the deployment is « kubernetes-bootcamp »).

img : kubernetes.io

Kubernetes : How to use - Example



```
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-v6wdx  1/1     Running   0          10m
$ kubectl get rs
NAME                             DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4   1         1         1       10m
```

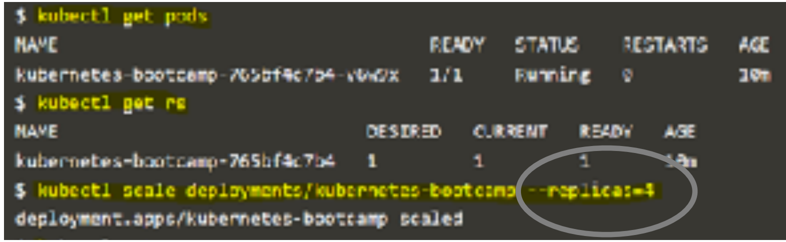
1 – list of pods on the cluster

2 – list of « *replicasets* »

The **replicaset** is a constraint applied to pods. If the number of pods don't match the desired number, it'll create or delete pods.
Here number of replicas is set to 1.

img : kubernetes.io

Kubernetes : How to use - Example



```
$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-v6w2x   1/1     Running   0          10m
$ kubectl get rs
NAME                             DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4   1         1         1       10m
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.apps/kubernetes-bootcamp scaled
```

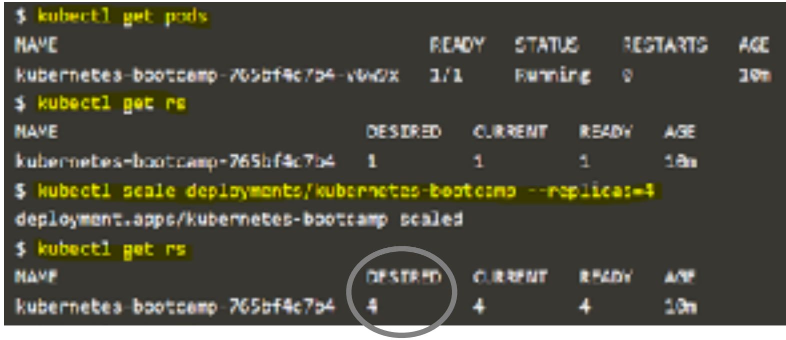
1 – list of pods on the cluster

2 – list of « *replicasets* »

3 – increase desired number to 4

img : kubernetes.io

Kubernetes : How to use - Example



```
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-v6w2x  1/1     Running   0          10m
$ kubectl get rs
NAME                            DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4  1         1         1       10m
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get rs
NAME                            DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4  4         4         4       10m
```

1 – list of pods on the cluster

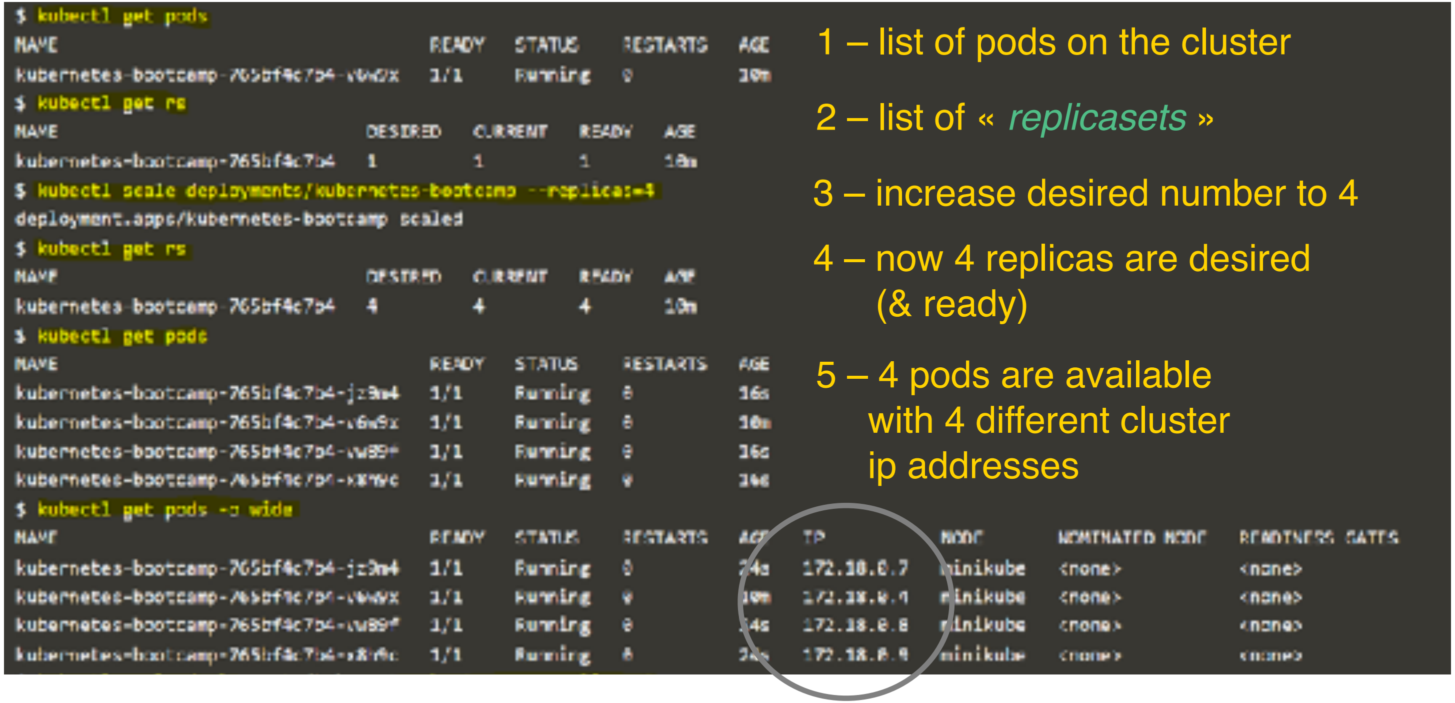
2 – list of « *replicasets* »

3 – increase desired number to 4

4 – now 4 replicas are desired
(& ready)

img : kubernetes.io

Kubernetes : How to use - Example



```
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-v6w2x   1/1     Running   0          10m
$ kubectl get rs
NAME                             DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4   1         1         1       10m
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get rs
NAME                             DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4   4         4         4       10m
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-jz9m4   1/1     Running   0          16s
kubernetes-bootcamp-765bf4c7b4-v6w9x   1/1     Running   0          10m
kubernetes-bootcamp-765bf4c7b4-vw89f   1/1     Running   0          16s
kubernetes-bootcamp-765bf4c7b4-x8r9c   1/1     Running   0          16s
$ kubectl get pods -o wide
NAME                                   READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-765bf4c7b4-jz9m4   1/1     Running   0          24s   172.18.0.7   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-v6w2x   1/1     Running   0          10m   172.18.0.4   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-vw89f   1/1     Running   0          24s   172.18.0.8   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-x8r9c   1/1     Running   0          24s   172.18.0.9   minikube   <none>           <none>
```

1 – list of pods on the cluster

2 – list of « *replicasets* »

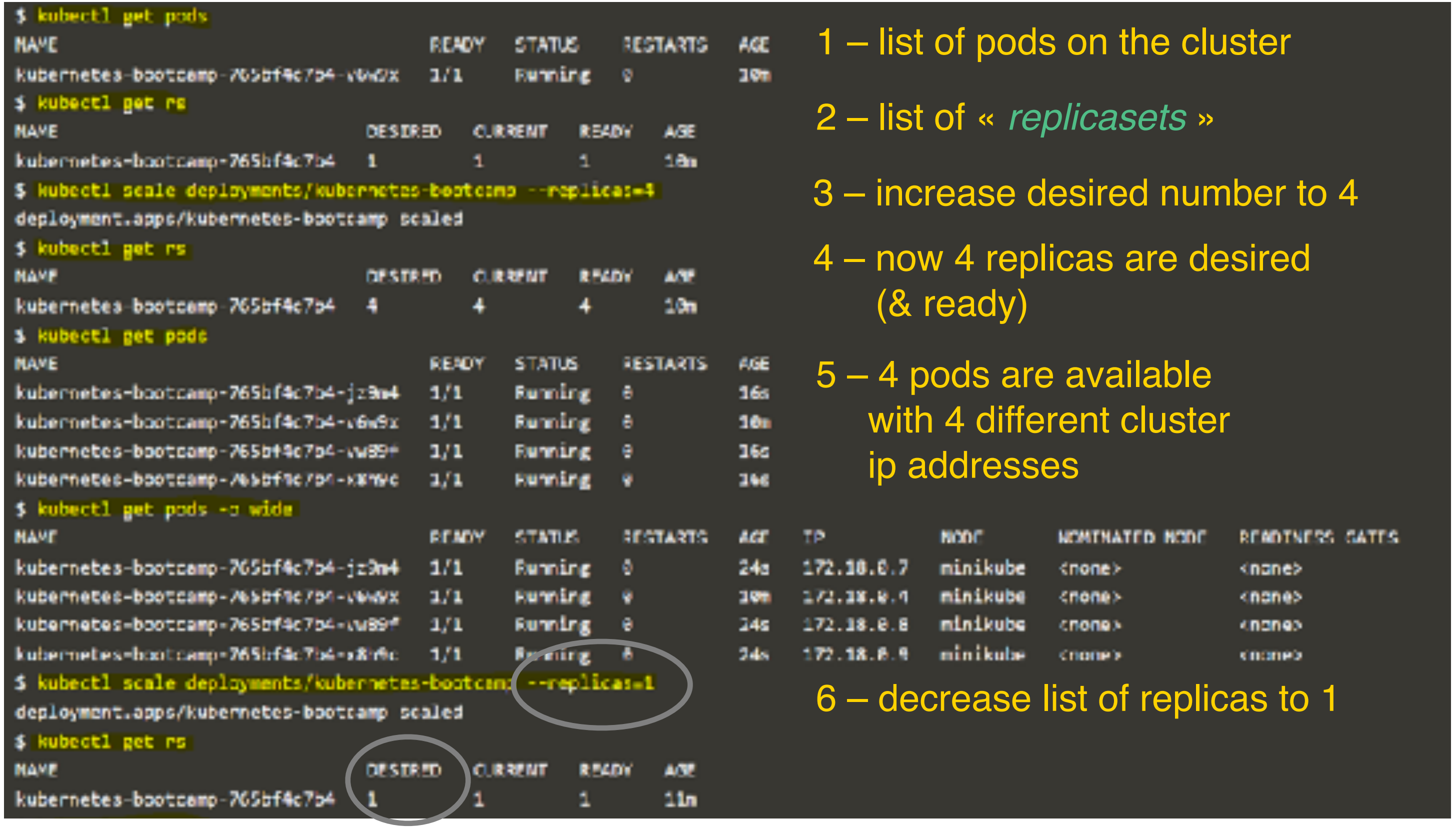
3 – increase desired number to 4

4 – now 4 replicas are desired
(& ready)

5 – 4 pods are available
with 4 different cluster
ip addresses

img : kubernetes.io

Kubernetes : How to use - Example



```
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-v6w2x   1/1     Running   0          10m
$ kubectl get rs
NAME                             DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4   1         1         1       10m
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get rs
NAME                             DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4   4         4         4       10m
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-jz9n4   1/1     Running   0          16s
kubernetes-bootcamp-765bf4c7b4-v6w9x   1/1     Running   0          10m
kubernetes-bootcamp-765bf4c7b4-wv89f   1/1     Running   0          16s
kubernetes-bootcamp-765bf4c7b4-x8rwc   1/1     Running   0          16s
$ kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-765bf4c7b4-jz9n4   1/1     Running   0          24s   172.18.0.7   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-v6w9x   1/1     Running   0          10m   172.18.0.4   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-wv89f   1/1     Running   0          24s   172.18.0.8   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-x8r6c   1/1     Running   0          24s   172.18.0.9   minikube   <none>           <none>
$ kubectl scale deployments/kubernetes-bootcamp --replicas=1
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get rs
NAME                             DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4   1         1         1       11m
```

1 – list of pods on the cluster

2 – list of « *replicasets* »

3 – increase desired number to 4

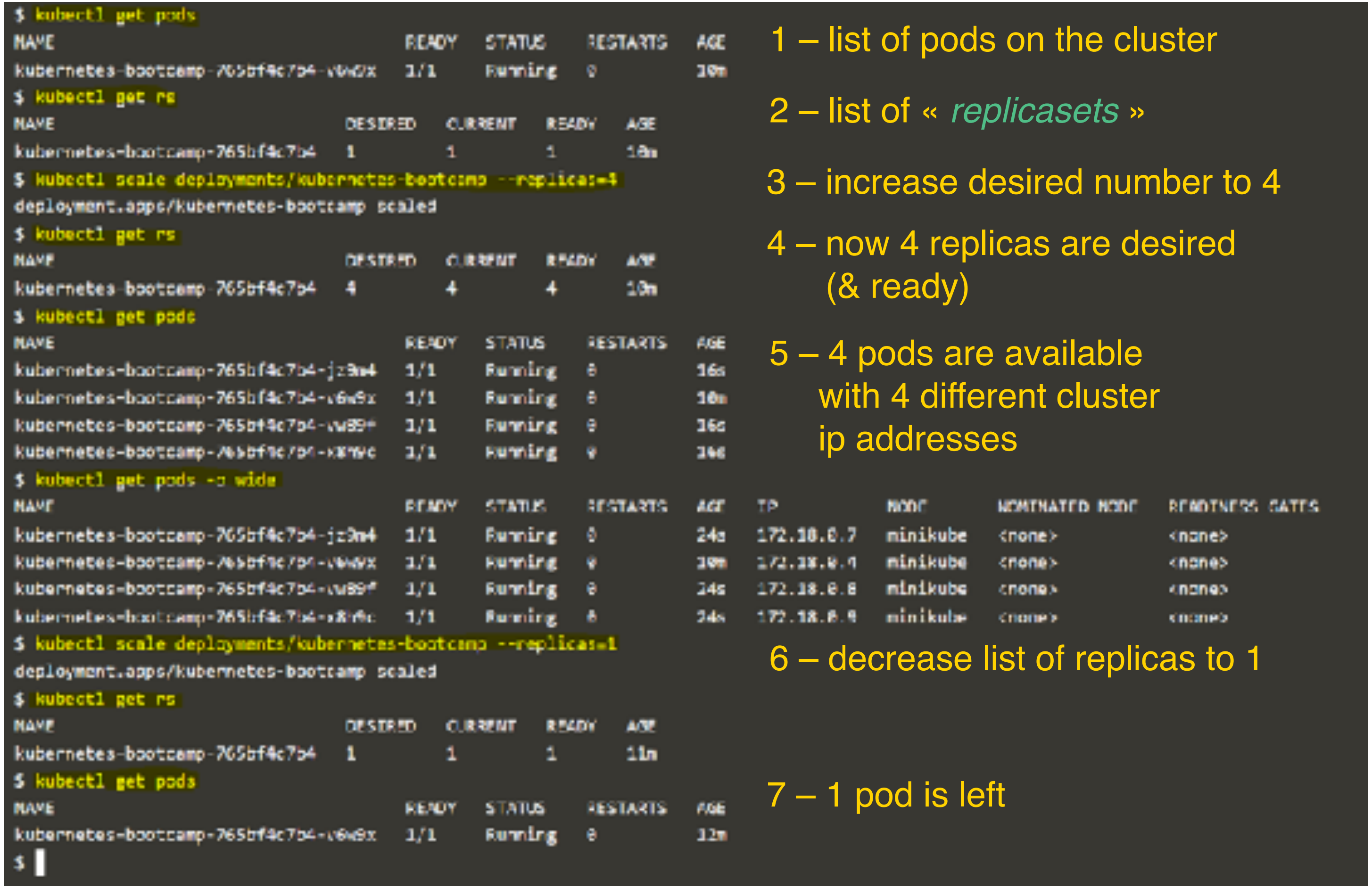
4 – now 4 replicas are desired
(& ready)

5 – 4 pods are available
with 4 different cluster
ip addresses

6 – decrease list of replicas to 1

img : kubernetes.io

Kubernetes : How to use - Example



```
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-v6w9x  1/1     Running   0          10m
$ kubectl get rs
NAME                            DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4  1         1         1       10m
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get rs
NAME                            DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4  4         4         4       10m
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-jz9m4  1/1     Running   0          16s
kubernetes-bootcamp-765bf4c7b4-v6w9x  1/1     Running   0          10m
kubernetes-bootcamp-765bf4c7b4-vw89f  1/1     Running   0          16s
kubernetes-bootcamp-765bf4c7b4-x8h9c  1/1     Running   0          16s
$ kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-765bf4c7b4-jz9m4  1/1     Running   0          24s   172.18.0.7   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-v6w9x  1/1     Running   0          10m   172.18.0.4   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-vw89f  1/1     Running   0          24s   172.18.0.8   minikube   <none>           <none>
kubernetes-bootcamp-765bf4c7b4-x8h9c  1/1     Running   0          24s   172.18.0.9   minikube   <none>           <none>
$ kubectl scale deployments/kubernetes-bootcamp --replicas=1
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get rs
NAME                            DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-765bf4c7b4  1         1         1       11m
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-765bf4c7b4-v6w9x  1/1     Running   0          12m
$ ▌
```

1 – list of pods on the cluster

2 – list of « *replicasets* »

3 – increase desired number to 4

4 – now 4 replicas are desired
(& ready)

5 – 4 pods are available
with 4 different cluster
ip addresses

6 – decrease list of replicas to 1

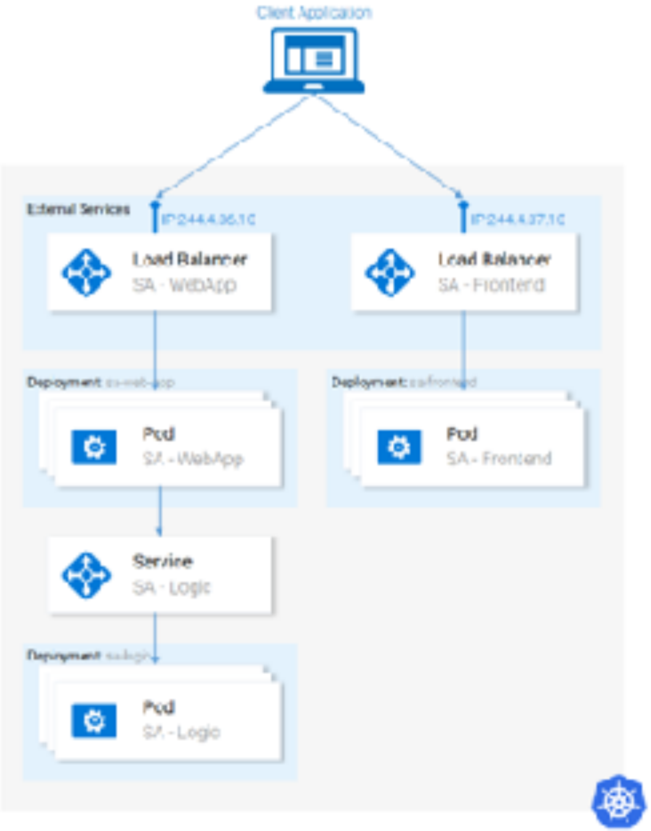
7 – 1 pod is left

img : kubernetes.io

Kubernetes : Other features

- Services : Load balancing
- Deployments : Zero-Downtime (BGD : Blue Green Deploiement) etc.
- Performance monitoring
- Use of yaml files to fully describes the way applications are deployed **(declarative configuration)**

« `kubectl apply -f /path/to/file.yaml` »



img : Medium tuto

Kubernetes : Declarative configuration principle

How to : deploy 3 replicas of a piece of software when only 1 is running

Declarative

« Set replica=3 »

Imperative

« Add 2 replicas »

Advantages

Rollback scenario :

- Apply config v2

** Bad things happened **

- Re-apply config v1

Self-healing algorithm :

- Do current state match desired state ?
  - Yes : re-check
  - No : re-apply desired state

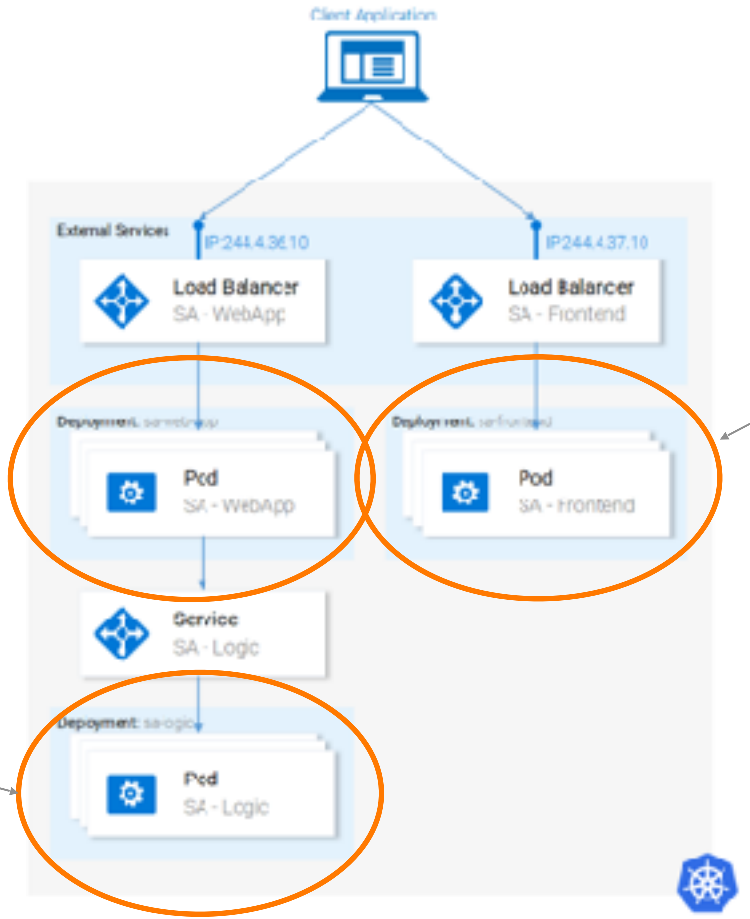
Controllers will check that the current state is identical to the desired state.

Nb: If you manually creates a 4th replica, kubernetes will destroy it.

Kubernetes : Microservices

Backend app:
 - Python functions
behind http

AI service behind URL :
Input : sentence
Output : sentiment analysis
(0 is bad, 1 is good)

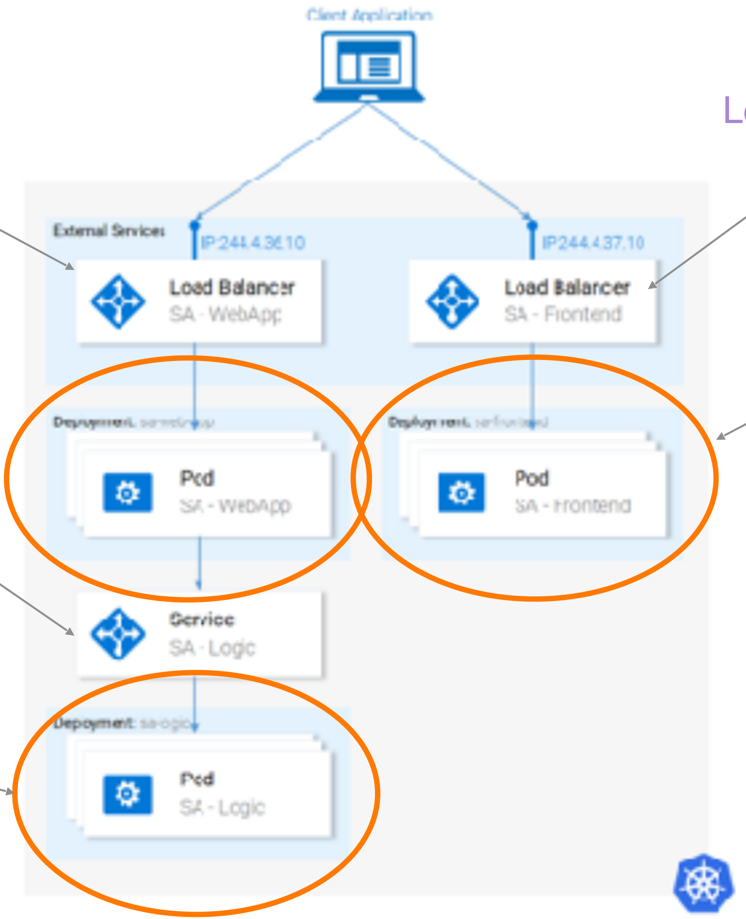


Frontend app:
- Nginx that serves
HTML/CSS/JS files

Then uses javascript to make http calls (from user's browser) to the backend to retrieve data

img : Medium tuto

Kubernetes : Microservices workload management



Load Balancer

Load Balancer

Replicated service

Replicated service

Load Balancer

Replicated service

** for high performance delivery of static content (html, css, js), use a CDN may be more efficient.*

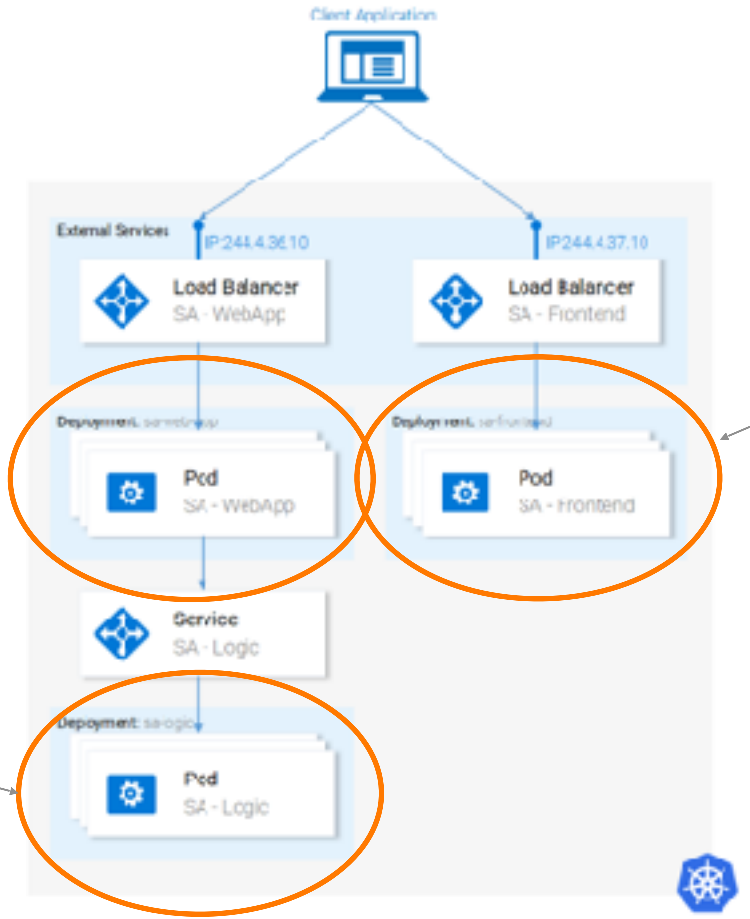
img : Medium tuto

Kubernetes : Microservices -> 1 service / team

BACKEND TEAM

FRONTEND TEAM

AI TEAM



img : Medium tuto

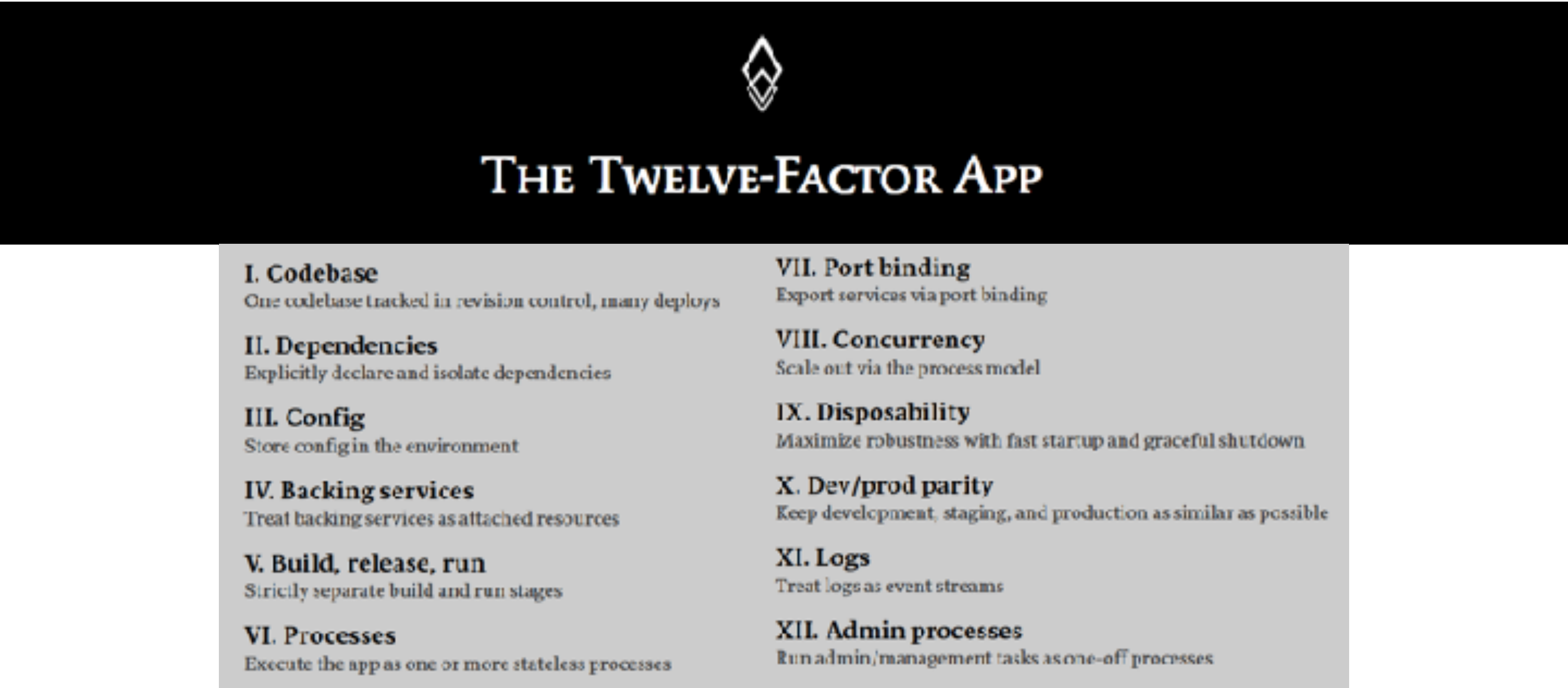
Kubernetes : Resources

- Nice example on Medium : https://medium.com/free-code-camp/learn-kubernetes-in-under-3-hours-a-detailed-guide-to-orchestrating-containers-114ff420e882

img : Medium tuto

Objective : 12 Factors app methodology

Methodology to build web apps.



THE TWELVE-FACTOR APP

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

https://12factor.net/

Merci